REMARKS

The claims remaining in the present application are Claims 1-23. Claims 1, 10, and 13-15 have been amended. Claims 17-23 have been added. No new matter has been added as a result of these amendments.

35 U.S.C. §102

Claims 1-16 are rejected under 35 U.S.C. §102(e) as being anticipated by Lethin et al., U.S. Patent Application Publication 2002/0,147,969 (hereinafter, Lethin). The rejection is respectfully traversed under the following rationale.

Embodiments of the present invention are concerned with determining whether a sequence of instructions should be interpreted or translated. In order to determine which of these operations should be applied to the instructions, an embodiment of the present invention tests at intervals whether an interpreter or a translator is operating. In various embodiments, the interval may be a time period or a number of executed instructions. The decision to interpret or translate is not tied to the number of times that a particular sequence of instructions is executed, in some embodiments of the present invention. Making the determination of whether to translate or interpret based on testing, at intervals, whether an interpreter or a translator is operating is non-intuitive and non-obvious.

In contrast, conventional methods typically count the number of times a particular instruction (e.g., a branch instruction) or a sequence of instructions is executed. If the particular instruction(s) is executed more than a given number of times, those particular instructions are then compiled (or translated) rather than interpreted.

Claim 1

Claim 1 recites:

> A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set comprising the steps of:
> executing code morphing software including an interpreter and a translator to generate host instructions from target instructions,
> detecting at intervals whether the interpreter or the translator is operating,
> increasing a count if the interpreter is operating and decreasing the count if the translator is operating, and
> changing from interpreting to translating a sequence of target instructions [[when]] in response to the count [[reaches]] reaching a selected maximum.

Claim 1 recites a limitation of "detecting at intervals whether the interpreter or the translator is operating." Detecting at intervals whether the interpreter or translator is operating involves executing the code morphing software for an interval, then determining whether the interpreter or the translator is operating, then executing the code morphing software for another interval, then determining whether the interpreter or the translator is operating, etc. Claim 1 recites an embodiment of the present invention in which the determination of whether to translate or interpret is based on detecting at intervals whether the interpreter or the translator is operating. This contrasts with a conventional techniques that merely base this decision on a count of the number of times a particular instruction(s) is executed.

Regarding the cited art, Lethin discloses a technique that follows the conventional technique of making a determination as to interpret or compile a sequence of instructions based on a count of the number of times an instruction (or instructions) is executed. Moreover, Lethin fails to teach or suggest the claimed limitation of detecting at intervals whether the interpreter or the translator is operating. Lethin uses feedback to adjust the threshold count that determines how many times an instruction(s) should be executed before the sequence should be translated instead of compiled. However, the decision as to whether to interpret or compile is based on a count of the number of times a

particular sequence of instruction is executed and has nothing to do with detecting at intervals whether the interpreter or translator has been operating. Moreover, adjusting the threshold dynamically does not teach or suggest, "detecting at intervals whether the interpreter or the translator is operating," as claimed.

The rejection asserts that Lethin at paragraphs 626-627 discloses Applicants' claimed limitation of detecting at intervals whether the interpreter or the translator is operating. However, these paragraphs describe counting how often a particular instruction(s) is executed and adjusting a threshold using feedback. Lethin begins paragraph 626 by stating that the interpreter counts how many times an instruction is executed. As Applicants have previously discussed, counting the number of times that a particular instruction(s) is executed is a conventional technique that is substantially different from the Applicants non-obvious technique of detecting at intervals whether the interpreter or the translator is operating, maintaining a count based thereon and changing from interpreting to translating a sequence of instructions based on that count.

The remainder of paragraph 626 discusses that Lethin is following the conventional technique of keeping track of execution counts, while adjusting the threshold that determines when instructions should be compiled rather than interpreted. For example, Lethin recites in paragraph 626 that, [w]hen the threshold is lower than most of the execution counts, the rate of translation request is too high. Moreover, adjusting the threshold does not constitute the claimed limitation of detecting at intervals whether the interpreter or the translator is operating.

Lethin goes on in paragraph 627 to describe a comparison procedure, which is the basis of adjusting the threshold. Lethin also describes this procedure as a software feedback technique for equalizing the rate of translation requests to the rate of

translations competed, in paragraph 625. Lethin's technique is to compare the number of translation requests sent to the translator to the number of translations completed. However, this comparison does not constitute the claimed limitation of "detecting at intervals whether the interpreter or the translator is operating." The claimed "detecting at an interval" detects that at the given interval, either the interpreter or the translator is being used. However, comparing one rate to another rate does not involve detection of which operation is then occurring. Moreover, Lethin's technique does not involve testing at an interval. For example, the comparison of one rate to another does not provide information as to whether, at a given point in the execution, the interpreter or translator is detected as being used. Rather, the comparison in Lethin is, "x requests for translation were made" and "y requests for translation were filled." All this comparison provides is a measure of a difference between the number of requests made and request filled. This comparison does not determine whether interpreting or translating is happening at a point defined by an interval, as claimed by Applicants.

Regarding Lethin's teaching, Applicants do understand that a request for translation occurs when the threshold is passed for a particular instruction(s). That is, those instructions have been executed a given number of times. However, the above comparison in Lethin does not provide any information as to whether the interpreter or the compiler was operating at a particular point as defined by the interval. Thus, this comparison fails to teach or suggest the claimed limitation of, "detecting at intervals whether the interpreter or the translator is operating, and based on a count of this, switching from interpreting to translating a sequence of instructions."

Claim 1 further recites the limitation of, "changing from interpreting to translating a sequence of target instructions in response to the count reaching a selected maximum." Applicants have amended Claim 1 to clarify that the changing is in response to the count

reaching a selected maximum. That is, the count (of interpreter usage minus translator usage) reaching a selected maximum is a trigger that initiates the change from interpreting to translating. Applicants note that the claimed count is based on interpreter utilization versus compiler utilization. Importantly, the count is not based on counting the number of times a sequence of instructions is executed. In contrast, Lethin discloses that the decision to change from interpreting to compiling a sequence of instructions is made in response to how many times the sequence of instructions has been executed. That is, when the threshold in Lethin is passed, indicating a sequence of instruction has been executed a given number of times, the mode is changed.

First, Lethin does not change from interpreting to compiling in response to the count (of interpreter to translator activity) reaching a selected maximum. That is, Lethin does not have a selected maximum for a count of interpreter to translator activity. Rather, Lethin has a threshold for the number of times that a particular sequence of instructions may be executed. In Lethin, the measure of interpreter to translator activity can reach any value without causing a switch to/from compiling/interpreting. It is only in response to the count of the number of times that a particular sequence of instructions has executed crossing the adjustable threshold that a switch from interpreting to translating occurs, in Lethin. Thus, Lethin does not disclose a selected maximum for the count (of interpreter to translator activity), as claimed.

Next, Lethin fails to make the switch to compiling in response to the count (of interpreter to translator activity) reaching a selected maximum. Rather, Lethin teaches switching in response to the threshold being crossed for the count of the number of times a particular sequence of instructions has been executed. This is because in Lethin a sequence of instructions must have been executed a number of times equal to the threshold in order to trigger the switch from interpreting to translating. Thus, Lethin does

not teach or suggest changing from interpreting to compiling in response to the count (of interpreter to translator activity) reaching a selected maximum.

For the foregoing rationale, it is respectfully submitted that Claim 1 is neither taught nor suggested by Lethin. As such, allowance of Claim 1 is respectfully submitted.

Claims 2-9 depend from Claim 1, which are respectfully believed to be allowable. As such, allowance of Claims 2-9 is earnestly solicited.

CLAIM 10

Currently Amended Independent Claim 10 recites:

> A method of optimizing execution by a computer which dynamically converts instructions from a target instruction set to a host instruction set comprising the steps of:
>
> providing a plurality of instruction conversion processes each providing a different level of optimization for converted instructions from a target instruction set to a host instruction set,
>
> providing means for determining dynamically which conversion process to use to convert each sequence of instructions, said means depending on detecting at intervals which of the instruction conversion processes is operating, and
>
> converting a sequence of instructions using a conversion process determined by said means to convert the sequence of instructions.

Claim 10 has been amended by removing the limitation of "best". Applicants have removed this limitation to more clearly describe this embodiment of the invention. Claim 10 has also been amended to recite that the means for determining dynamically which conversion process to use to convert each sequence of instructions depends on detecting at intervals which instruction conversion process is operating.

For the reasons discussed in the response to Claim 1, it is respectfully asserted that Lethin fails to teach or suggest the above limitation of Claim 10. For the foregoing rationale, it is respectfully submitted that Claim 10 is neither taught nor suggested by Lethin. As such, allowance of Claim 10 is respectfully submitted.

Claims 11-15 depend from Claim 10, which are respectfully believed to be allowable. As such, allowance of Claims 11-15 is earnestly solicited.

Claim 16 recites, in part:

> comparing interpreter usage to translator usage when executing said code morphing software to produce an interpreter usage to translator usage factor, and
>
> changing from interpreting to translating a sequence of target instructions if the interpreter usage to translator usage factor crosses a threshold.

Claim 16 recites that *if* the interpreter usage to translator usage factor crosses a threshold, a change is made from interpreting to translating a sequence of target instructions. Applicants respectfully assert that Lethin fails to teach or suggest this claim limitation. Thus, Claim 16 recites a cause and effect condition. That is, if the usage factor crosses a threshold, then the method of handling the instructions is changed.

Lethin does not teach or suggest the claimed cause and effect condition. Lethin fails to make the switch from interpreting to compiling if the interpreter usage to translator usage factor crosses a threshold. This is because in Lethin a sequence of instructions must have been executed a number of times equal to the threshold in order to trigger the switch from interpreting to translating. Thus, in Lethin, the cause of a change from interpreting to compiling is the crossing of the adjustable threshold for a count of the

number of times that a particular sequence of instructions is executed. Thus, Lethin does not teach or suggest changing from interpreting to compiling if the interpreter usage to translator usage factor crosses a threshold.

Moreover, Lethin does not change from interpreting to compiling if the interpreter usage to translator usage factor *crosses a threshold*. This is because Lethin does not disclose a threshold for an interpreter usage to translator usage factor. As previously discussed, the measure of interpreter to translator activity in Lethin can reach any value without causing a switch to/from compiling/interpreting. It is only in response to the count of times that a particular sequence of instruction is executed crosses the adjustable threshold of instruction usage that a switch from interpreting to translating occurs. For the foregoing, Lethin does not disclose a threshold for an interpreter usage to translator usage factor, as claimed.

For the foregoing reasons, Claim 16 is neither taught nor suggested by Lethin. Consequently, Applicants respectfully request allowance of Claim 16.

New Claims

Claims 17-23 have been added. Support for these claims may be found in the instant specification at least from page 11, line 10 - page 15, line 16.

Claim 17 recites:

> A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set, said method comprising:
>
> generating host instructions from target instructions by executing software comprising an interpreter and a translator;
>
> during said generating, detecting at intervals whether said interpreter or said translator is operating;

in response to said detecting, increasing a count if the interpreter is operating and decreasing said count if said translator is operating;

in response to said count reaching a selected maximum, recording a sequence of instructions then being processed by said generating; and

changing from interpreting to translating said sequence of instructions upon said sequence of instructions next being encountered by said generating (emphases added).

Claim 17 recites, "during generating of host instructions, detecting at intervals whether an interpreter or a translator is operating." Claim 17 also recites "in response to said count reaching a selected maximum, recording a sequence of instructions then being processed by said generating, and changing from interpreting to translating said sequence of instructions upon said sequence of instructions next being encountered by said generating." For at least the reasons discussed in the response to Claim 1, Claim 17 is believed to be allowable over the cited art. Claims 18-23 depend from Claim 17 and are believed to be allowable as a consequence of this dependency.

CONCLUSION

In light of the above listed amendments and remarks, reconsideration of the rejected Claims is requested. Based on the arguments and amendments presented above, it is respectfully submitted that Claims 1-23 overcome the rejections of record and, therefore, allowance of Claims 1-23 is earnestly solicited. Should the Examiner have a question regarding the instant amendment and response, the Applicants invite the Examiner to contact the Applicants' undersigned representative at the below listed telephone number.

Respectfully submitted,
WAGNER, MURABITO & HAO LLP

Dated: 2/3, 2004

Ronald M. Pomerenke
Registration No. 43,009

Address: WAGNER, MURABITO & HAO LLP
Two North Market Street
Third Floor
San Jose, California 95113

Telephone: (408) 938-9060 Voice
(408) 938-9069 FAX